Giving Digital Life to a Print Dictionary

Michael Maxwell & Aric Bills
University of Maryland
mmaxwell/abills@umd.edu

University of Maryland, College Park MD 20742 USA

February 2019

Resurrecting a print dictionary

- We used OCR and parsing technology, to convert a print dictionary into a database.
- An *experiment*!
  - Build and test methodology for converting OCR output to dictionary database
  - *Not* in focus: imaging pages of print dictionary, OCR technology
- We tested the resulting database by using it to build a parser, and then parsing texts.
- Were we successful?
- BLUF: It's not as easy as it seems!
- Why do this?
  - Old print dictionaries often exist where no electronic dictionaries exist
  - Even if there is a modern dictionary, the old dictionary (and texts) may represent a more vibrant stage of the language

How did we do it?

1. Run an OCR engine on a PDF-image dictionary

How did we do it?

1. Run an OCR engine on a PDF-image dictionary
2. Convert verbose output of OCR into lines of text in columns, with indentation represented as “tab” stops (inferred by clustering)

How did we do it?

1. Run an OCR engine on a PDF-image dictionary
2. Convert verbose output of OCR into lines of text in columns, with indentation represented as “tab” stops (inferred by clustering)
3. ...cleanup, e.g. convert ligatures to sequences of characters

How did we do it?

1. Run an OCR engine on a PDF-image dictionary
2. Convert verbose output of OCR into lines of text in columns, with indentation represented as “tab” stops (inferred by clustering)
3. ...cleanup, e.g. convert ligatures to sequences of characters
4. ...omit unreliable information (bolding, italics)

How did we do it?

1. Run an OCR engine on a PDF-image dictionary
2. Convert verbose output of OCR into lines of text in columns, with indentation represented as “tab” stops (inferred by clustering)
3. ...cleanup, e.g. convert ligatures to sequences of characters
4. ...omit unreliable information (bolding, italics)
5. Convert multiple lines of texts representing a single entry into a single line (lines not starting with “tab” start a new entry; combine across column and page breaks)

How did we do it?

1. Run an OCR engine on a PDF-image dictionary
2. Convert verbose output of OCR into lines of text in columns, with indentation represented as “tab” stops (inferred by clustering)
3. ...cleanup, e.g. convert ligatures to sequences of characters
4. ...omit unreliable information (bolding, italics)
5. Convert multiple lines of texts representing a single entry into a single line (lines not starting with “tab” start a new entry; combine across column and page breaks)
6. “Typo” correction: simple string replacement using manually written fixup rules

How did we do it?

1. Run an OCR engine on a PDF-image dictionary
2. Convert verbose output of OCR into lines of text in columns, with indentation represented as “tab” stops (inferred by clustering)
3. ...cleanup, e.g. convert ligatures to sequences of characters
4. ...omit unreliable information (bolding, italics)
5. Convert multiple lines of texts representing a single entry into a single line (lines not starting with “tab” start a new entry; combine across column and page breaks)
6. “Typo” correction: simple string replacement using manually written fixup rules
7. Rule-driven parsing of text lines to XML entries

How did we do it?

1. Run an OCR engine on a PDF-image dictionary
2. Convert verbose output of OCR into lines of text in columns, with indentation represented as “tab” stops (inferred by clustering)
3. ...cleanup, e.g. convert ligatures to sequences of characters
4. ...omit unreliable information (bolding, italics)
5. Convert multiple lines of texts representing a single entry into a single line (lines not starting with “tab” start a new entry; combine across column and page breaks)
6. “Typo” correction: simple string replacement using manually written fixup rules
7. Rule-driven parsing of text lines to XML entries
8. Debug and iterate. And iterate. And iterate...

The Process

- Earlier description of process in Maxwell and Bills 2017
- Input: SIL's 1999 edition of the *Diccionario Tzeltal de Bachajón, Chiapas*, compiled by Marianna Slocum, Florence Gerdel, and Manuel Cruz Aguilar (Cruz, Gerdel, and Slocum 1999); about 150 pages Tzeltal -> Spanish
- Parsing engine: Tesseract (https://github.com/tesseract-ocr)
- Output: hOCR = XHTML-based representation of text, formatting, and position on page
- We used Tesseract's Spanish OCR model
- Also referred to Slocum 1948 grammar

The Dictionary

jts'o'sit *s* ciego

jts'umbajom *s* principal

jucaw *s* iniciador

jucawan *vi* iniciar, incitar
ya xjucawan lo inicia

jucbayel ta carabato *vt* coger con garabato
la sjucbay ta carabato cogió con garabato

jucul q'uinal *s* valle

juch' *vn* acto de moler **Och ta juch' ta cha' te antse.** La mujer empezó a moler en un metate.

juch'el *vt* moler
juch'bil está molido
juch'bil cacaw chocolate
la sjuch' lo molió

jujun habil cada año
jujutuhl cada persona
jujuts'ihn a cada rato, a menudo
ta jujun uno por uno

julawal *adj* agudo, penetrante
julawal ch'ix una espina aguda

julbac *s* dolencia reumática, reúma *Véase* **bac**

julel 1. *vt* punzar (*p. ej.: espina*)
2. *vt* inyectar
3. *s* inyección
la sjul lo inyectó
la yich' julel lo inyectaron

julub *s* formón (*herramienta*)

jun *adj num* uno
jun habil un año
jun semana una semana

Sample hOCR data

```
...
<div class='ocr_page' id='page_1' title='image \
    "/tmp/tmp94gf7qd_.png"; bbox 0 0 2550 3300; \
    ppageno 0'>
...
<div class='ocr_carea' id='block_1_7'
    title="bbox 519 1033 1026 1191">
<p class='ocr_par' dir='ltr' id='par_1_7'
    title="bbox 519 1033 1026 1191">
 <span class='ocr_line' id='line_1_11' \
       title="bbox 519 1033 970 1073; \
       baseline 0.002 -9; x_size 39; \
       x_descenders 8; x_ascenders 10">
 <span class='ocrx_word' id='word_1_32'
       title='bbox 519 1034 587 1064; x_wconf 89'
       lang='eng' dir='ltr'>ahb</span>
...
```

Simplify hOCR output

- Input: hOCR files, one per page
- Infer lines of text within a column by reference to bounding boxes of each “span”
- Throw away page headers and footers (and deal with section headers)
- Infer tab stops of each line by clustering of indents relative to inferred left margins of column(s) (2–4 clusters = tab stops, manually selected)
- Omit extraneous information (bounding boxes, bolding, italicization)
- Output: Simplified representation of OCR output, with indents represented by `tab` characters
- (Keyed-in input could start here)

Simplified representation

aba pron refl te, se (2.“ pers.)
[pl.: abahic]
tehc’ana aba párate
tsahtaya me abahic cuidense

abac s suciedad, tizne
la yich’ abac se puso sucio

ac’a –uc part Indica deseo o
mandato en modo subjuntivo;
p. ej.: ¡ac’a taluc! ¡que venga!

Infer lexical entries

- Infer lexical entries: a line that doesn't begin with a tab stop is a new entry (N.B.: Tesseract's inferred paragraphs were *not* helpful.)
- Replace `tab` characters with a visible tag
- Combine parts of lexical entries across column and page breaks
- Output: Lines of text with embedded tags for tab-based indentation

One lexical entry per line representation

```
aba pron refl te, se (2.“ pers.) \
[pl.: abahic]<2INDENT>tehc’ana aba \
párate<2INDENT>tsahtaya me abahic cuidense

abac s suciedad, tizne<2INDENT>la yich’ \
abac se puso sucio

ac’a –uc part Indica deseo o mandato en \
modo subjuntivo; p. ej.: ¡ac’a taluc! \
¡que venga!
```

Typo Correction

- Typos are (usually) single character confusions:
  - 1 l, 0 O or o (digit letter)
  - Missing or hallucinated accents
  - i ¡, l →]
  - Quotation marks
- Sometimes a very simple correction: ‘].’ should always be ‘1.’
- …but careful: entry-initial ‘1’ should almost always be ‘l’…*except* when it should be a homograph number!
- We could have simply edited output of OCR
- We chose instead to have simple string replacement rules:

  lbat s corcho (árbol)

  1bat s corcho (árbol)
- This allows re-running upstream processes

Parsing

- Output: an XML database
  - Could have used a standard (like LBX)
  - ...but settled for something simple
- Use finite state transducer (FST) for conversion
- Rather than use the parsing engine's rule notation, we invented one specifically for dictionary parsing, and translated it into the programming language of the FST
- Example rule (~40 of these):

```
Entry <Entry> = Headwords OMIT(<Space>+)
    ( ( ( (POSs (OMIT(<Space>+) Etymology)?)
        | Variant
        )
        OMIT(<Space>+) Senses (OMIT(<Space>+)
            IrregularForm)*
    )
    | SubMultiEntries
    )
    (OMIT(<2INDENT>) SubEntry)*;
```

After extracting the XML dictionary, we tested it by building a *morphological* parser

- Parser building process is described elsewhere (David and Maxwell 2008; Maxwell 2010, 2012, 2013, 2015a,b; Maxwell and David 2008), but briefly:
  - Create an XML-based declarative representation of the grammar
  - Automatically convert that into Stuttgart FST code
  - SFST parsing engine imports grammar + information extracted from XML dictionary
  - Test parser by parsing texts

What could possibly go wrong?

- Initial attempts at parsing dictionary gave dismal results: few dictionary entries parsed
- Reasons:
  - Typos
  - Unexpected (inconsistent) entry structure
  - Most frequent words often had complex dictionary entries, and/or most inconsistencies wrt. other entries
  - Lack of reliable tagging of bold text made parsing sub-entries unreliable
- Solutions:
  - Debug: error due to typo or rule?
  - Add the typo correction, fix the rule, or add a new rule
  - Run the parsing process again, and hope
- Some dictionary entries parsed in multiple ways, usually because of inability to know where headwords of sub-entry stopped, and gloss began (but also sometimes because our rules were ambiguous)

Debugging the dictionary parser

- Early failures led us to develop debugging tools. Most important information:
  - What was the furthest point the parser got to on an entry?
  - What was it expecting to see at that point?
  - What structure was it building?

```
ac'a -uc part Indica deseo o mandato en modo
subjuntivo; p. ej.: ¡^ac'a taluc! ¡que venga!
--partial parse:
<Entry><Headwords><FormRep lang="tzh">ac'a -uc
</FormRep></Headwords><POSs><POS>part </POS></POSs>
<Senses><Sense><Explanation>deseo o mandato en
modo subjuntivo </Explanation><Example>
<ExS lang="tzh">¡
parser expects: ['A','B','C','D','E','F','G','H','I',
'J','L','M','N','O','P','Q','R','S','T','U','V','W',
'X','Y']
```

Eventually judged the dictionary “good enough”

- 4743 dictionary entries total
- 4401 entries parse
- 342 entries fail to parse
- average parse ambiguity 1.15
- (We made a few further fixes based on errors in morphological parsing of texts)

Morphological parsing of texts

- Is the XML dictionary good enough for text parsing?
- We found (a few) texts on-line: http://www.indigenouspeople.net/frontera/conjuradatzet.htm
  - About 5000 words
  - ...in variable orthography
  - ...with dialectal(?) variation
- Tokenized on white space and punctuation (except apostrophe)
- Did some cleanup of non-ASCII characters.
- First result: 330 parsed words out of 4963 tokens :-(
- Later result (due to repairs on next slide): 1900 parsed words :-|
- Many more fixes are possible, but we have a proof of concept.

Attacked improvements starting with high frequency items

- Used Linux command-line tools to sort parse failures by frequency, etc.
- Four monosyllabic words accounted for 940 of the failed parses (programming error in dictionary import)
- Next most common failure was entry for definite article `te...-e`, representing obligatory agreement between definite article and noun (two entries like this)
- Failure to parse two complex dictionary entries for homograph `yu'un` (= “their”, “because”)
- Forgot to remove citation form suffix from verbs
- Past tense particle `la`: discussed in dictionary’s appendix, but no lexical entry; created “entry” in grammar

...More improvements based on high frequency items

- Orthography changes: ‘c’/‘qu’ → ‘k’: There was a spelling rule for this, but part of its environment was missing
- Unexpected orthography difference: ‘tz’ for ‘ts’
- Dialectal variation:
  - /x/ (‘j’) contrasts with /h/ (‘h’) in dictionary dialect, but contrast is lost in some dialects; merged phoneme always spelled ‘j’ in one of the documents.
  - Definite article ‘te’ consistently spelled ‘ti’ in one of the documents

Lessons learned

- Converting a print dictionary to electronic form is feasible
- It's just not a easy as we had hoped.

Lessons learned

- Converting a print dictionary to electronic form is feasible
- It's just not a easy as we had hoped.
- Monkey wrenches:
  - OCR is not as good as we had hoped

Lessons learned

- Converting a print dictionary to electronic form is feasible
- It's just not a easy as we had hoped.
- Monkey wrenches:
  - OCR is not as good as we had hoped
  - Inconsistencies in dictionary

Lessons learned

- Converting a print dictionary to electronic form is feasible
- It's just not a easy as we had hoped.
- Monkey wrenches:
  - OCR is not as good as we had hoped
  - Inconsistencies in dictionary
  - Orthography changes

Lessons learned

- Converting a print dictionary to electronic form is feasible
- It's just not a easy as we had hoped.
- Monkey wrenches:
  - OCR is not as good as we had hoped
  - Inconsistencies in dictionary
  - Orthography changes
  - Dialectal variation

Lessons learned

- Converting a print dictionary to electronic form is feasible
- It's just not a easy as we had hoped.
- Monkey wrenches:
  - OCR is not as good as we had hoped
  - Inconsistencies in dictionary
  - Orthography changes
  - Dialectal variation
- To make it easier:
  - Do OCR better
  - ...including bolding/italics (if relevant)

Lessons learned

- Converting a print dictionary to electronic form is feasible
- It's just not a easy as we had hoped.
- Monkey wrenches:
  - OCR is not as good as we had hoped
  - Inconsistencies in dictionary
  - Orthography changes
  - Dialectal variation
- To make it easier:
  - Do OCR better
  - ...including bolding/italics (if relevant)
  - If some information is not needed, just parse it in as text (Example: morphological parsers don't need sense information, or even subentries)

Lessons learned

- Converting a print dictionary to electronic form is feasible
- It's just not a easy as we had hoped.
- Monkey wrenches:
  - OCR is not as good as we had hoped
  - Inconsistencies in dictionary
  - Orthography changes
  - Dialectal variation
- To make it easier:
  - Do OCR better
  - ...including bolding/italics (if relevant)
  - If some information is not needed, just parse it in as text (Example: morphological parsers don't need sense information, or even subentries)
  - Use morphological parsing (or your favorite application) to look for problems in high frequency entries

Lessons learned

- Converting a print dictionary to electronic form is feasible
- It's just not a easy as we had hoped.
- Monkey wrenches:
  - OCR is not as good as we had hoped
  - Inconsistencies in dictionary
  - Orthography changes
  - Dialectal variation
- To make it easier:
  - Do OCR better
  - ...including bolding/italics (if relevant)
  - If some information is not needed, just parse it in as text (Example: morphological parsers don't need sense information, or even subentries)
  - Use morphological parsing (or your favorite application) to look for problems in high frequency entries
- Expectation setting.

Contact information

Do~~n't~~ try this at home!

Michael Maxwell and Aric Bills
University of Maryland
{mmaxwell/abills}@umd.edu

With funding by the National Science Foundation under grant number BCS1644606.

Citations I

Cruz, Manuel A., Florence L. Gerdel, and Marianna C. Slocum (1999). *Diccionario tzeltal de Bachajón, Chiapas*. Serie de vocabularios y diccionarios indígenas "Mariano Silva y Aceves" 40. Coyoacán, D.F., Mexico: Instituto Lingüístico de Verano, A.C. URL: https://www.sil.org/resources/archives/10969.

David, Anne and Michael Maxwell (2008). "Joint Grammar Development by Linguists and Computer Scientists." In: *IJCNLP*. Hyderabad: The Association for Computer Linguistics, pp. 27–34. URL: http://aclweb.org/anthology/I/I08/I08-3007.pdf.

Maxwell, Michael (2010). "Standardization as a means to sustainability." In: *Workshop on Language Resources: From Storyboard to Sustainability and LR Lifecycle Management*. LREC 2010. Malta, pp. 30–33. URL: http://www.lrec-conf.org/proceedings/lrec2010/workshops/W20.pdf.

– (2012). "Electronic Grammars and Reproducible Research." In: *Electronic Grammaticography*. Ed. by Sebastian Nordoff and Karl-Ludwig G. Poggeman. University of Hawaii Press, pp. 207–235. URL: http://scholarspace.manoa.hawaii.edu/handle/10125/4536.

Citations II

Maxwell, Michael (2013). "A System for Archivable Grammar Documentation." In: *Systems and Frameworks for Computational Morphology: Proceedings of the Third International Workshop on Systems and Frameworks for Computational Morphology*. Ed. by Cerstin Mahlow and Michael Piotrowski. Switzerland: Springer, pp. 72–91.

– (2015a). "Accounting for Allomorphy in Finite State Transducers." In: *Proceedings of the the 12th International Conference on Finite-State Methods and Natural Language Processing (FSMNLP 2015)*. Ed. by Thomas Hanneforth. Vol. 537. Communications in Computer and Information Science (CCIS). URL: https://aclweb.org/anthology/W/W15/W15-4809.pdf.

– (2015b). "Grammar Debugging." In: *Systems and Frameworks for Computational Morphology: Fourth International Workshop, SFCM 2015*. Ed. by Cerstin Mahlow and Michael Piotrowski. Vol. 537. Communications in Computer and Information Science, pp. 166–183.

Citations III

Maxwell, Michael and Aric Bills (2017). "Endangered Data for Endangered Languages: Digitizing Print dictionaries." In: *Proceedings of the 2nd Workshop on the Use of Computational Methods in the Study of Endangered Languages*. Honolulu: Association for Computational Linguistics, pp. 85–91. URL: http://www.aclweb.org/anthology/W17-0112.

Maxwell, Michael and Anne David (2008). "Interoperable Grammars." In: *First International Conference on Global Interoperability for Language Resources (ICGL 2008)*. Ed. by Jonathan Webster, Nancy Ide, and Alex Chengyu Fang. Hong Kong, pp. 155–162. URL: http://hdl.handle.net/1903/11611.

Slocum, Marianna C. (1948). "Tzeltal (Mayan) Noun and Verb Morphology." In: *International Journal of American Linguistics* 14.2, pp. 77–86. ISSN: 00207071, 15457001. URL: http://www.jstor.org/stable/1263231.